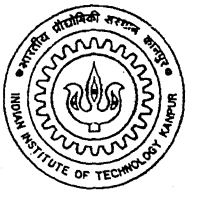
Providing Evidence for Robustness of the Universality Notion

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Jugal Garg



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

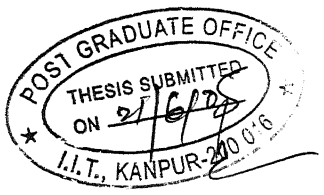
June, 2005

Certificate

This is to certify that the work contained in the thesis entitled "*Providing Evidence for Robustness of the Universality Notion*", by *Jugal Garg*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

June, 2005

(Dr. Somenath Biswas)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.



Abstract

We provide evidence for the robustness of the notion of universality (introduced by [AB92] for the class $\mathcal{NP}$). The natural NP-relations that define certain NP-complete problems like Graph k-Coloring are not immediately amenable to the formalism of [AB92] for proving their universality. In this thesis, we prove universality of such languages. In particular for the NP-complete problems: Graph k-Coloring (for $k \geq 3$), k-NM Colorability (for $k \geq 3$), Learning of k-DNF formulae (for $k \geq 3$), and Generic NP-complete problem. Later, we extend this notion to the complexity class 1-NL. 1-NL is the class of languages accepted by non-deterministic turing machines having one-way read-only input tape and logarithmic worktape. We define 1-NL languages in terms of 1-NL relations and then give a characterization of 1-NL universality. Finally, we present our work on PCP, which gives a different account for the class $\mathcal{NP}$ and defines a very different natural NP-relations. We sketch the proof of the PCP theorem $\mathcal{NP} = \text{PCP}(\log n, 1)$, which will later help us in proving the relation witnessing 3SAT as per PCP to be universal.

Acknowledgements

I take this immense opportunity to express my sincere gratitude towards my supervisor Dr. Somenath Biswas, for his guidance, invaluable suggestions, and constant support which made my thesis successful. I consider myself extremely fortunate to have had a chance to work under his supervision. Inspite of his hectic schedule he was always approachable and took his time off to attend to my problems and give the appropriate advice. It has been a very enlightening and enjoyable experience to work under him.

I am thankful to Dr. Manindra Agrawal for a discussion on PCP. I am also thankful to Chandan Dubey, Madhur Tulsiani, and Nitesh Kumar for their valuable support in the study of PCP.

Thanks to Dr. Milind Sohoni at IIT-Bombay who got me interested in theory and for his constant encouragement and support.

I also wish to thank whole heartily all the faculty members of the Department of Computer Science and Engineering for the invaluable knowledge they have imparted to me and for teaching the principles in most exciting and enjoyable way. I also extend my thanks to the technical staff of the department for maintaining an excellent working facility.

Finally, I thank my parents for always being there for me, for bringing me to this point in life.

Contents

List of Figures

Chapter 1

Introduction

The notion of universality was introduced by Agrawal and Biswas ([AB92]) for the class $\mathcal{NP}$, where they observed the structural similarities in natural NP-complete sets and found that the usual reductions among them are solution preserving, viz., when a language L_1 reduces to an NP-complete language L_2 via such a solution preserving polynomial-time many-one reduction f, then we can easily extract the solutions of the instance $x \in L_1$ from the solutions of the corresponding instance $f(x) \in L_2$. A solution for an instance is the witness that the instance belongs to a set S. Every $\mathcal{NP}$ set S can be seen to be associated with an NP-relation R_S, R_S being $\{(x, y) \mid x \in S$ and y a witness that shows x to be in $S\}$. A relation R_S for S is called universal if S is NP-complete under solution preserving reductions. Finally, [AB92] also characterized universal relations and gave an easy way to prove a set to be NP-complete by showing the two properties (joinability and couplability) hold for R_S and one particular instance called building block exists. Some relations witnessing natural NP-complete problems are not universal by the above characterization, so [AB92] generalized the universality notion to *near-universal*, and showed that these relations are near-universal.

Further, [AB92] proved several relations witnessing natural NP-complete problems to be universal by showing them joinable, couplable and having a building block.

Some proofs are easy to construct than the ones which are obtained using usual reductions. The Graph Isomorphism problem is not believed to be NP-complete and [AB92] gave further evidence by showing the standard relation witnessing it to be not near-universal. Apart from the universality proofs of several relations witnessing natural NP-complete problems, [AB92] also proved that the relations witnessing non-natural sets like *k-creative sets* ([JY85]) are universal.

1.1 Thesis

In this thesis, we prove the relations witnessing certain NP-complete problems to be universal in order to give evidence for the robustness of universality notion. Later, we extend this notion to the complexity class 1-NL and finally, we present our attempt towards PCP-universality.

1.1.1 Graph 3-Coloring

Graph 3-Coloring is a natural NP-complete graph problem, but the usual notion of couplability in its standard witnessing relation is not definable here due to the following reason. In most of the graph problems, a solution string encodes either vertices or edges indicating their presence or absence in the witness. Therefore, the meaning of coupling two bits in their corresponding relations is to enforce the condition that exactly one of the two vertices (or edges) represented by these two bits is present in the solution string. In the case of Graph 3-Coloring, a natural way to encode three colors is by using two bits for each color in the solution string. Therefore, coupling of any two bits in the solution string does not make any sense here. The natural meaning should be to couple the colors of two vertices, but according to the definition of couplability, it has to be a function that can couple any two bits in the solution string.

In chapter 2, we modify the definition of couplability for the relation witnessing

Graph 3-Coloring problem to show that this relation is near-universal. In the similar way, we can obtain the proof of universality for the standard relation witnessing Graph k-Coloring problem for any $k \geq 3$. Next, we show the universality of the relations witnessing some other problems including k-NM colorability (for $k \geq 3$), Learning of k-DNF formulae (for $k \geq 3$), and Generic NP-complete language.

1.1.2 1-NL

The notion of universality has also been defined in the similar fashion for the complexity classes $\mathcal{NL}$ ([CSB04]), and $\#\mathcal{P}$ ([CK03, FM03]). In chapter 3, we define this notion for the class 1-NL.

1-NL (1-L) is the class of languages accepted by non-deterministic (deterministic) Turing machines having one-way read-only input tape and logarithmic worktape. The classes L and NL are different from these classes, where the machines have two-way read-only input tape. 1-NL is computationally more powerful than 1-L and it is interesting to note that although it is computationally very weak than NL, yet any complete set for it under 1-L reductions is also complete for NL under L reductions ([HM81]). We know from the Immerman-Szelepcsényi result that NL is closed under complementation ([Imm88]), however 1-NL is not closed under complementation ([HU67]).

Like NL in [CSB04], we define 1-NL languages in terms of relations and then characterize 1-NL universal relations. The notion of universality for 1-NL gives a good understanding of it. This class is important because if a 1-NL complete problem can be shown in L, then the space bounded deterministic and non-deterministic computations for space $\Theta(\log n)$ collapses (i.e. L=NL).

1.1.3 PCP

PCP stands for Probabilistically Checkable Proofs. It denotes the class of languages accepted by PCP machines, which consist of a probabilistic polynomial time

verifier (a machine) with a read-only input tape (input is on this tape), a random tape (contains random bits used by the verifier), a proof tape (contains proof string provided by the prover for the input) and a work tape (the verifier does computations on this tape). PCP machines are classified by two resources, the number of random bits used by the verifier and the number of query bits read from the proof string. A verifier is said to be $(r(n), q(n))$-restricted if it uses $O(r(n))$ random bits and queries $O(q(n))$ bits from the proof string, and a language L belongs to the class PCP$(r(n), q(n))$, if the membership proofs for L can be checked by an $(r(n), q(n))$-restricted verifier. According to this classification, NP contains exactly those languages for which membership proofs can be checked by a $(\log n, 1)$-restricted verifier. The PCP theorem states the equivalence of NP and PCP$(\log n, 1)$ (for further details, see [Aro94, AS92, RS95, DK00, ALM+92, Sud92, Kum05]).

PCP gives a different account for the class NP as it defines a very different natural NP-relation. Therefore, it would be interesting to see whether the technique of *universal relations* defined for the classical NP-relations is also applicable to the NP-relations defined as per PCP. In chapter 4, we analyze the proof string through the sketch of the PCP theorem in order to understand the new definition of NP and later show that the relation witnessing 3-SAT as per PCP is universal.

1.2 Definitions and Important Results

These definitions and results are directly taken from [AB92] and the relations in these definitions are assumed to be NP-relations.

Definition 1.1 Function f, f: $\Sigma^* \rightarrow \Sigma^*$, is a *solution preserving reduction* of relation Q to relation R if it is polynomial-time computable and satisfies the following conditions.

1. $f(x) = \langle z, \alpha \rangle$ where $x, z \in \Sigma^*$ and α is a sequence with $|\alpha| = sol\text{-}len_Q(x)$.
2. $proj_\alpha(sol_R(z)) = sol_Q(x)$.

)efinition 1.2 Relation R is a *universal* relation if for every relation Q, there is a)lution-preserving reduction of Q to R.

'roposition 1.1 *If R is a universal relation then L_R is NP-complete under polynomial-ime honest reductions.*

)efinition 1.3 3-SAT: Given a formula ψ, where ψ is conjunction of clauses, and ach clause has three literals, the problem 3SAT is to determine whether ψ is satis-iable.

The relation R_{3SAT} is: $\langle\psi\rangle R_{3SAT}\langle s\rangle$ iff s is a satisfying truth assignment for ψ.

?roposition 1.2 *Relation R_{3SAT} is universal.*

1.2.1 Characterization of Universal Relations

Definition 1.4 Relation R has a *building block* if there is an element in L_R, $block_R$, and three positive integers bit_1, bit_2, and bit_3 such that

$$proj_{bit_1,bit_2,bit_3}(sol_R(block_R)) = \Sigma^3 - \{000\}.$$

Definition 1.5 Relation R is *joinable* if there exists a polynomial-time computable function $join_R$, $join_R : \Sigma^* \to \Sigma^*$, satisfying the following conditions.

1. $join_R(\langle x_1, ..., x_n\rangle) = \langle z, \alpha\rangle$ where $x_1, ..., x_n, z \in \Sigma^*$ and $|\alpha| = \Sigma_{k=1}^{n} sol\text{-}len_R(x_k)$.
2. $proj_\alpha(sol_R(z)) = \{s_1 s_2 \cdots s_n \mid (\forall k \leq n) s_k \in sol_R(x_k)\}$.

Definition 1.6 Relation R is *couplable* if there exists a polynomial-time computable function cpl_R, $cpl_R : \Sigma^* \to \Sigma^*$, satisfying the following conditions.

1. $cpl_R(x, \langle i_1, ..., i_n\rangle, \langle j_1, ..., j_n\rangle) = \langle z, \alpha\rangle$ where $x \in \Sigma^*, 1 \leq i_1, ..., i_n, j_1, ..., j_n \leq sol\text{-}len_R(x)$, $i_m \neq j_m$ for each $1 \leq m \leq n$ and $|\alpha| = sol\text{-}len_R(x)$.
2. $proj_\alpha(sol_R(z)) = \{s | s \in sol_R(x)$ and $(\forall m \leq n)\ s[i_m] \neq s[j_m]\}$.

Proposition 1.3 *Relation R is universal iff R is joinable, couplable and has a building block.*

2 Near-Universality

ition 1.7 Relation $\tilde{R}$ is a *refinement* of the relation R if there is a polynomial-computable many-one function h, $h : \Sigma^* \to \Sigma^*$, such that for every x and s, on h satisfies the following conditions.

For every s, $h^{-1}(s)$ is either $\emptyset$ or contains s.

$x\tilde{R}h(s) \iff xRs$.

ition 1.8 Relation R is near-universal if there is a refinement $\tilde{R}$ of R such $\tilde{R}$ is universal.

osition 1.4 *Set S has a near-universal relation iff it has a universal relation.*

Chapter 2

NP Universality Results

As we stated in the introduction, the usual notion of couplability does not make sense for the standard witnessing relation of Graph 3-Coloring. A solution string of the Graph 3-Coloring instance is an encoding of colors assigned to all the vertices in the graph and a natural way to encode three colors is by using two bits for each color. Therefore, coupling of any two bits in the solution string does not make any sense here. The natural meaning should be to couple the colors of two vertices, but according to the definition of couplability, it has to be a function that can couple any two bits in the solution string.

In this chapter, we modify the definition of couplability for the relation witnessing Graph 3-Coloring problem to show that this relation is near-universal. In the similar way, we can obtain the proof of universality for the standard relation witnessing Graph k-Coloring problem for any $k \geq 3$. Next, we show the universality of the relations witnessing some other problems including k-NM colorability (for $k \geq 3$), Learning of k-DNF formulae (for $k \geq 3$), and Generic NP-complete language.

2.1 Graph 3-Coloring

First we show that the standard relation witnessing Graph 3-Coloring problem is near-universal and then apply the same technique to Graph k-Coloring for all $k \geq 3$.

Definition 2.1 Graph 3-Coloring: Given a graph $G = (V, E)$. The problem is to tell whether G is 3-colorable, that is, does there exist a function $f : V \rightarrow \{1, 2, 3\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$.

The relation R_{G3C} is: $\langle G \rangle R_{G3C} \langle s \rangle$ iff s encodes the colors assigned to vertices of G such that the above property satisfies. Without loss of generality, we assume that the three colors $\{1, 2, 3\}$ are encoded in the following binary strings $\{11, 01, 00\}$ respectively. For example, let $G = (V, E)$ and $V = \langle v_1, v_2, ..., v_n \rangle$, then $s = c(v_1)c(v_2) \cdots c(v_n)$, where $c(v_i)$ is the color encoding assigned to the vertex v_i (i.e. $c : V \rightarrow \{11, 00, 01\}$).

Theorem 2.1 *Relation R_{G3C} is near-universal.*

Proof: Notice that R_{G3C} has many redundant solutions. We say that two solutions are same if we can map the colors in one solution to the colors in another solution by a one-to-one mapping $\sigma : \{1, 2, 3\} \rightarrow \{1, 2, 3\}$. Therefore, a solution can be represented by six different colorings, and hence six different solution strings.

First we refine this relation to remove the above redundancy in the solution strings to some extent. In the refined relation, first vertex of the graph is always assigned color 1, which has encoding '11' in the solution string. Therefore, every solution string in the refined relation starts with 1 in its MSB position. Note that this refinement reduces many redundant solutions, however not completely, but now there are only two different solution strings representing the same solution. Let $\tilde{R}_{G3C}$ is the relation after refinement. Now we show the solution preserving reduction from R_{3SAT} to $\tilde{R}_{G3C}$.

We follow the reduction from 3SAT to Graph 3-Coloring as given in [CLR90]. From the given formula ψ of m clauses and n variables $\{x_1, x_2, ..., x_n\}$, we construct a graph $G = (V, E)$ as follows: The set V consists of a vertex for each variable, a vertex for negation of each variable, five vertices for each clause and three special vertices. These three special vertices are labeled as *true*, *false*, and *red*. The edges

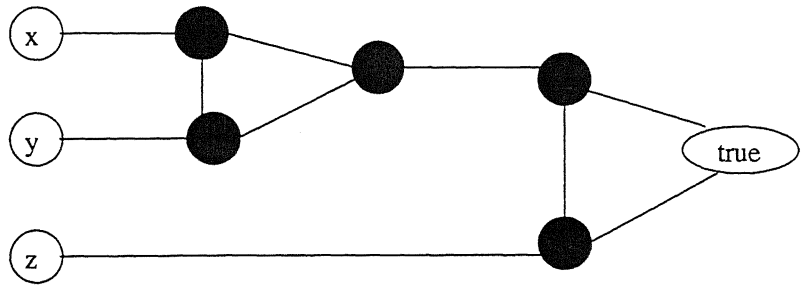


Figure 2.1: The widget corresponding to a clause $(x \vee y \vee z)$

the graph are of two types: one independent of the clauses and one dependent the clauses. The independent of the clauses edges form a triangle on the special rtices and also form a triangle on $x_i, \neg x_i$, and *red* for $1 \leq i \leq n$.

ıe triangle on the special vertices enforces the assignment of different colors to ch of the special vertex and the triangle on $x_i, \neg x_i$, and *red* for $1 \leq i \leq n$ enforces at exactly one of the variable and its negation is colored with $c(true)$, other is lored with $c(false)$, and they can not be colored with $c(red)$. The idea is that the riables, whose corresponding vertices are colored with $c(true)$, will get value 1 and e remaining variables will get value 0 in the truth assignment of the formula ψ. ı enforce the conditions corresponding to the clauses, for each clause, we have five tra vertices and they are connected in such a way, so that at least one literal of e clause will get value 1 (for satisfying the clause). For example, the construction widget corresponding to the clause $(x \vee y \vee z)$ is shown in figure 2.1.

ıus for every instance ψ of 3SAT, we have an instance $G = (V, E)$ of Graph 3-)loring such that for every truth assignment of ψ, there is a 3-Coloring of G.

ıe solutions of 3SAT instance can be obtained from the solutions of corresponding :aph 3-Coloring instance as follows:

ithout loss of generality, we assume that G is outputed as follows: the initially

the three special vertices $(true, false, red)$, then n vertices corresponding to the n variables, then n vertices corresponding to the negation of variables, then vertices corresponding to the m clauses, i.e. $V = \langle true, false, red, x_1, ..., x_n, \neg x_1, ..., \neg x_n, ... \rangle$, and finally all the edges of G. Observe that, the *true* labeled vertex is outputed as the first vertex in the graph, so in every solution, it is assigned the color encoded as '11'. The solutions of Graph 3-Coloring instance are encodings of colors assigned to each vertex and the solution length is $2 * |V|$. The projection sequence is just the sequence of the first bits of the colors assigned to the vertices corresponding to the variables (positive literal), i.e. $\alpha = 7, 9, ..., 2n + 5$ and $|\alpha| = n$.

Therefore, the variables, whose corresponding vertices are colored with $c(true)$, are assigned value 1, and the remaining variables, whose corresponding vertices are colored with $c(false)$, are assigned value 0 regardless of the color encoding given to $c(false)$, because the first bit of both the color encodings $\{00, 01\}$ is 0. This assignment satisfies the formula ψ, which is trivial to see from the reduction.

As we already know that R_{3SAT} is universal, therefore, $\tilde{R}_{G3C}$ is universal from the above solution preserving reduction. Hence, R_{G3C} is near-universal. ∎

Theorem 2.2 *Relation $\tilde{R}_{G3C}$ is universal using characterization of universal relations ([AB92]).*

Proof: We prove that $\tilde{R}_{G3C}$ is universal by showing that $\tilde{R}_{G3C}$ is joinable, couplable and has a building block. We assume that the graph $G = (V, E)$ has n vertices, and they are labeled as numbers from 1 to n.

For the building block, the graph $G = (V, E)$, where $V = \langle 1, 2, ..., 13 \rangle$ is given in figure 2.2. The solution string s consists of the color encodings assigned to all the vertices in the same order as given in V (i.e. $s = c(1)c(2) \cdots c(13)$). As each color encoding has 2 bits, so $|s| = 26$. Next, we show that taking $bit_1 = 5$, $bit_2 = 9$, and $bit_3 = 13$ (i.e. first bits of the color encodings to the vertices 3, 5, and 7) as in the definition of *building block* satisfies the required properties.

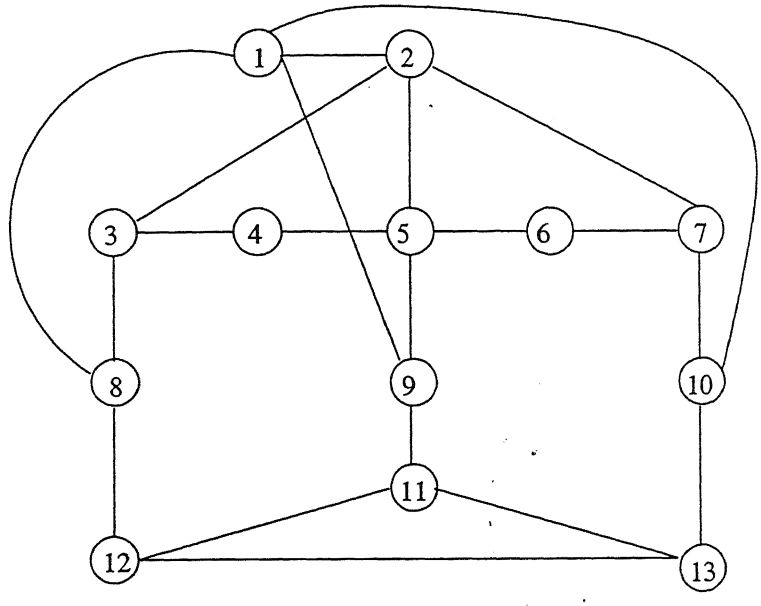


Figure 2.2: Graph for the instance $block_{\tilde{R}_{G3C}}$

Since the vertex 1 is always assigned color '11' and there is an edge between 1 and 2, so 2 can be colored with either '00' or '01', let first fix it to be '00' ('01' also gives the same result). The vertex 2 is also connected with 3, 5 and 7, so these three vertices can be colored with either '11' or '01'. The vertices 8, 9, and 10 are connected with 1, so they can be colored with either '00' or '01', and since they are also connected with 12, 11, and 13 respectively and there is a triangle on 12, 11, and 13, so all three vertices 8, 9 and 10 can not be given the same color. The vertices 3, 5, and 7 are also connected with 8, 9, and 10 respectively, therefore, all three vertices 3, 5, and 7 cannot be assigned the same color '01'. So if we take the first bits of colors assigned to 3, 5, and 7, then we will get all three bits strings except '000' (i.e. $\Sigma^3 - \{000\}$).

For joinability, we define $bprod_{\tilde{R}_{G3C}}(x, y) = \langle z, \alpha \rangle$ where z is obtained as follows: Let x has n vertices and y has m vertices, then the vertices of y are renumbered

by adding n to their numbers. We add two new dummy vertices (labeling them $n+m+1$ and $n+m+2$) and a triangle on vertex 1 and these dummy vertices. This will ensure that the dummy vertices are assigned color encodings $\{00, 01\}$. Now, we add another triangle on vertex $n+1$ and these dummy vertices. This will ensure that the vertex $n+1$ is assigned color encoding '11'.

The projection α is the sequence of all the bits of the solution of z except the last 4 bits, which correspond to the color encodings assigned to the dummy vertices.

For couplability, we show that $\tilde{R}_{G3C}$ is couplable in a restricted sense, however it is sufficient for our purpose of proving the relation to be universal. Note that, the length of every solution of $\tilde{R}_{G3C}$ is always even and the bits of projection (bit_1, bit_2, and bit_3) of building block are odd, and when we join several building blocks to form an instance of Graph 3-Coloring corresponding to any arbitrary 3SAT instance in the proof of Theorem 4.4 of [AB92], then also every bit of projection is odd. Later on the same proof, all the coupling is done among these bits only, so if we restrict the couplability to only odd bits of the solution, then that is sufficient for showing the relation universal.

Define,

$$cpl_{\tilde{R}_{G3C}}(x,i,j) = \begin{cases} < z, \alpha > & : \text{ where both } i \text{ and } j \text{ are odd numbers and } i \neq j \\ undefined & : \text{ otherwise} \end{cases}$$

Let x has n vertices given as $\langle v_1, v_2, ..., v_n \rangle$. Note that i^{th} and j^{th} bits correspond to the first bits of the color encodings assigned to vertex $v_{\lceil i/2 \rceil}$ and $v_{\lceil j/2 \rceil}$. z and α are obtained as follows:

- If one of i and j is 1, then put an edge between $v_{\lceil i/2 \rceil}$ and $v_{\lceil j/2 \rceil}$. In this case, α is identity.
- If both i and j are not equal to 1, then add a new dummy vertex $n+1$, then add an edge between v_1 and v_{n+1}, and then add a triangle on $v_{\lceil i/2 \rceil}$, $v_{\lceil j/2 \rceil}$

and v_{n+1}. In this case, α is all the bits of the solution except the last 2 bits corresponding to the color encoding of dummy vertex.

', it is easy to see that all required properties are satisfied. ∎

Graph k-Coloring

nition 2.2 Graph k-Coloring: Given a graph $G = (V, E)$. The problem is to vhether G is k-colorable, that is, does there exist a function $f : V \rightarrow \{1, 2, ..., k\}$ that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$.

relation R_{GkC} is: $\langle G \rangle R_{GkC} \langle s \rangle$ iff s encodes the colors assigned to vertices of that the above property satisfies. Without loss of generality, we assume that : colors are encoded in the binary strings of length $\lceil \log k \rceil$.

orem 2.3 *Relation R_{GkC} is near-universal for $k \geq 3$.*

of: This proof is a little modification of the earlier proof for Graph 3-Coloring. also, we refine the relation R_{GkC} to $\tilde{R}_{GkC}$ as follows:

$_{kC}$, the first vertex of the graph is always assigned the color encoded as $0 \cdots 0$ all bits are 0), and the next different colored vertex in the vertex ordering of raph is always assigned the color encoded as $0 \cdots 01$ (i.e. only LSB bit is 1). every color encoding has length $\lceil \log k \rceil$. For example, let $V = \langle v_1, v_2, ..., v_n \rangle$. ding to our assignment, v_1 is always assigned color encoded as $0 \cdots 0$, and let of v_1 is same as v_j for $2 \leq j \leq i$, and v_{i+1} has to be assigned different color he colors of all the previous vertices in the vertex ordering, so it is assigned ncoded as $0 \cdots 01$.

e show the solution preserving reduction from R_{3SAT} to $\tilde{R}_{GkC}$.

the given formula ψ of m clauses and n variables $\{x_1, x_2, ..., x_n\}$, we con-
a graph $G = (V, E)$ as follows: The set V consist of a vertex for each variable,
ex for negation of each variable, five vertices for each clause and k special
ɔs. The k special vertices are labeled as $true, false, sp_1, ..., sp_{k-2}$. The edges
graph are of two types: one independent of the clauses and one dependent
clauses. The independent of the clauses edges form a complete graph on the
l vertices and also form a complete graph on $x_i, \neg x_i, sp_1, ..., sp_{k-2}$ for $1 \leq i \leq n$.

ɔmplete graph on special vertices enforces the assignment of different col-
each of the special vertex and the complete graph on $x_i, \neg x_i, sp_1, ..., sp_{k-2}$ for
$\leq n$ enforces that exactly one of the variable and its negation is colored with
) $(c : V \rightarrow \{1, 2, ..., k\})$, other is colored with $c(false)$, and they can not be
l with $c(sp_i)$ for all $1 \leq i \leq (k-2)$. The idea is that the variables, whose
ɔonding vertices are colored with $c(true)$, will get value 1 and the remaining
es will get value 0 in the truth assignment of the formula ψ. To enforce the
ons corresponding to the clauses, for each clause, we have five extra vertices
ey are connected in such a way so that at least one literal of the clause will
ue 1 (for satisfying the clause). For example, the construction of widget cor-
ding to the clause $(x \vee y \vee z)$ is shown in figure 2.3. There, the five extra
ɜ labeled as $1, 2, ..., 5$ can only be assigned colors from $c(true)$, $c(false)$, and
in order to meet the required conditions.

ɔr every instance ψ of 3SAT, we have an instance $G = (V, E)$ of Graph k-
g such that for every truth assignment of ψ, there is a k-Coloring of the graph.

utions of 3SAT instance can be obtained from the solutions of corresponding
k-Coloring instance as follows:

t loss of generality, we assume that the graph is outputed as follows: initially
ıl vertices $(false, true, sp_1, ..., sp_{k-2})$, then n vertices corresponding to the
ɔles, then n vertices corresponding to the negation of variables, and finally
corresponding to the m clauses, i.e. $V = \langle false, true, sp_1, sp_2, ..., sp_{k-2}, x_1,$

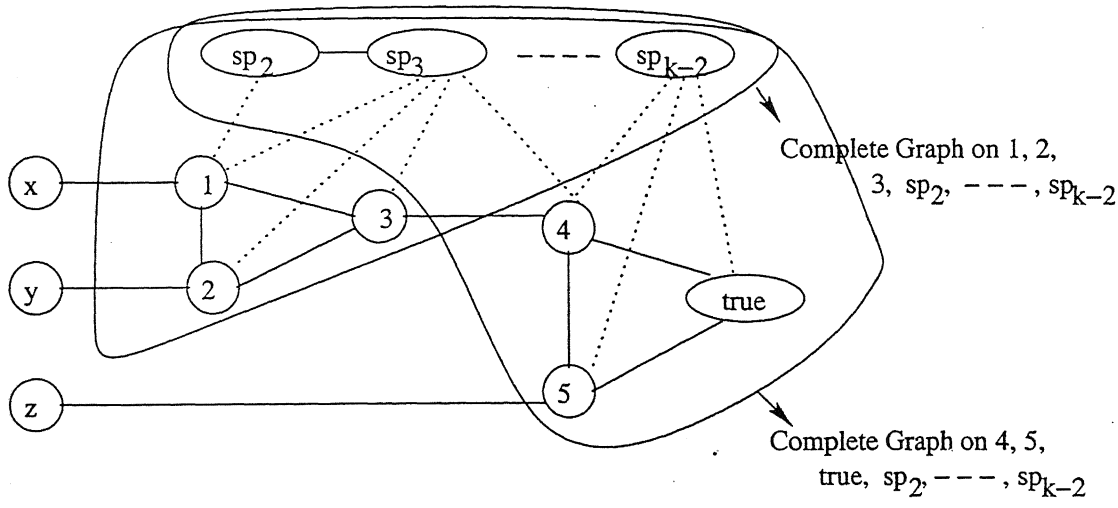


Figure 2.3: The widget corresponding to a clause $(x \vee y \vee z)$

$..., x_n, \neg x_1, ..., \neg x_n, ...\rangle$. Note that the *false* labeled vertex is outputed as the first vertex in the graph, so in every solution, it will be assigned the color encoded as $0 \cdots 0$ (all bits are 0), and the *true* labeled vertex is outputed as the second vertex in the graph, so in every solution, it will be assigned the color encoded as $0 \cdots 01$ (only LSB bit is 1) because there is a complete graph on all special vertices. The solutions of Graph k-Coloring instance are encodings of colors assigned to every vertex and their length is $\lceil \log k \rceil * |V|$. The projection sequence is just the sequence of the last bits of the colors assigned to the vertices corresponding to the variables (positive literals), i.e. $\alpha = (k+1)*\lceil \log k \rceil, (k+2)*\lceil \log k \rceil, ..., (k+n)*\lceil \log k \rceil$ and $|\alpha| = n$.

Therefore, the variables, whose corresponding vertices are colored with $c(true)$, will get value 1 and others will get value 0, and this assignment satisfies the formula ψ, which is trivial to see from the reduction.

As we already know that R_{3SAT} is universal, therefore, for $k \geq 3$, $\tilde{R}_{GkC}$ is also universal from the above solution preserving reduction. Hence, R_{GkC} is near-universal for $k \geq 3$. ∎

Proposition 2.1 *For $k \geq 3$, relation $\tilde{R}_{GkC}$ is universal using characterization of universal relations ([AB92]).*

Proof: The proof can be easily constructed in the similar way as we did for $k = 3$ in theorem 2.2. ∎

2.3 k-NM Colorability

Definition 2.3 k-NM colorability: Given a finite set S and a collection $C = \{c_1, c_2, ..., c_m\}$ of constraints $c_i \subseteq S$. The problem is to find if there is a k-Coloring of the elements of S (i.e. a function $\xi : S \rightarrow \{1, 2, ..., k\}$), such that for each constraint $c_i \in C$, the elements of c_i is Not Monochromatically colored (i.e. $(\forall c_i \in C)$ $(\exists x, y \in c_i)$ such that $\xi(x) \neq \xi(y)$).

The relation R_{kNMC} is: $\langle S, C \rangle R_{kNMC} \langle s \rangle$ iff s encodes the colors assigned to elements of S, such that the above property satisfies. Without loss of generality, we assume that the k colors are encoded in the binary strings of length $\lceil \log k \rceil$.

Definition 2.4 Set Splitting problem: The input is a collection C of subsets of finite set S and the problem is to decide whether there is a partition of S into two subsets S_1 and S_2 such that no subset in C is entirely contained in either S_1 or S_2.

Lemma 2.4 *For all integers $k \geq 2$, k-NM colorability is NP-complete.*

Proof: Clearly, k-NM colorability is in NP. For $k \geq 3$, there is a direct reduction from Graph k-Coloring to it. Now we only need to show NP-hardness for $k = 2$. 2-NM colorability is same as Set Splitting problem, which is also known to be an NP-complete problem ([GJ79]). ∎

Theorem 2.5 *Relation R_{kNMC} is near universal for $k \geq 3$.*

Proof: Like we did in R_{GkC}, we refine this relation R_{kNMC} to $\tilde{R}_{kNMC}$ as follows: To the first variable of the set S, we always assign the color encoded as $0 \cdot \cdot 0$ (all

bits are 0), and to the second different colored variable in the variable ordering, we always assign color encoded as $0\cdots01$ (only LSB bit is 1). Notice that this refinement eliminates some redundant solutions.

Clearly, there is a solution preserving reduction from $\tilde{R}_{GkC}$ to $\tilde{R}_{kNMC}$ for $k \geq 3$. Since $\tilde{R}_{GkC}$ is universal, therefore, $\tilde{R}_{kNMC}$ is also universal. Hence, R_{kNMC} is near universal for $k \geq 3$. ∎

2.4 Learning k-term DNF Formulae

Definition 2.5 Learning k-term-DNF formulae: Given a set of positive and negative examples, the problem is to output a k-term-DNF formula, which is consistent with the given set of examples.

The relation R_{kDNF} is: $\langle S \rangle R_{kDNF} \langle \psi \rangle$, where ψ is a k-term-DNF formula consistent with S.

Definition 2.6 Learning monotone k-term-DNF formulae: Given a set of positive and negative examples, the problem is to output a monotone k-term-DNF formula, which is consistent with the given set of examples.

The relation R_{k-mDNF} is: $\langle S \rangle R_{k-mDNF} \langle \psi \rangle$, where ψ is a monotone k-term-DNF formula consistent with S.

Lemma 2.6 *For $k \geq 2$, Learning k-term-DNF formulae is NP-Complete.*

Proof: Learning k-term-DNF formulae is clearly in NP. For NP-hardness, we can reduce k-NM colorability to the learning problem of k-term-DNF formulae ([PV88]). Therefore, it is NP-complete. ∎

Corollary 2.7 *For $k \geq 2$, Learning monotone k-term-DNF formulae is NP-complete.*

Proof: Clearly, learning monotone k-term-DNF formulae is in NP. For NP-ıardness, we can reduce learning problem of k-term-DNF formulae to it ([KV94]). Therefore, it is NP-complete. ∎

Next, we will show that R_{kDNF} is near universal for $k \geq 3$. For that, we need ı different solution representation in R_{GkC}. First, we prove that different definiion of R_{GkC} is also near-universal. After that, we will similarly define different olution representation in R_{kNMC} and prove it to be near-universal, and then we vill prove R_{k-mDNF} to be near-universal. In the end, we will show that R_{kDNF} is .ear-universal for $k \geq 3$.

We modify the solution representation of Graph k-Coloring as follows. In the soıtion string, every color is encoded as k bits string, where i^{th} bit is 0 implies that olor i can be assigned. Notice that, now an encoded color can represent more than ne possible color assignment to the corresponding variable.

emma 2.8 *The modified relation R_{GkC} is near-universal for $k \geq 3$.*

roof: First we refine this modified relation R_{GkC} to modified $\tilde{R}_{GkC}$ to eliminate ɔme redundant solutions as follows: first vertex of the graph is always assigned ɔlor encoded as $01\cdots1$ (only MSB bit is 0). Note that, now every solution string arts with 0 in its MSB position.

ow, we can show solution preserving reduction from R_{3SAT} to modified $\tilde{R}_{GkC}$. he whole proof is same as proved earlier for the different solution representation. herefore, modified R_{GkC} is near-universal. ∎

ext, we modify the solution representation of k-NM Colorability as follows. In e solution string, every color is encoded as k bits string, where i^{th} bit is 0 implies at color i can be assigned.

mma 2.9 *The modified relation R_{kNMC} is near-universal for $k \geq 3$.*

Proof: First we refine this modified relation R_{kNMC} to modified $\tilde{R}_{kNMC}$ as follows: first variable is always assigned color encoded as $01 \cdot \cdot 1$ (only MSB bit is 0). Therefore, now every solution string starts with 0.

Clearly, there is a solution preserving reduction from modified $\tilde{R}_{GkC}$ to modified $\tilde{R}_{kNMC}$. Therefore, modified $\tilde{R}_{k-NMC}$ is universal and modified R_{kNMC} is near-universal. ∎

Theorem 2.10 *Relation R_{k-mDNF} is near-universal for $k \geq 3$.*

Proof: The solution string in R_{k-mDNF} is represented by nk bits, where n is the number of variables in the input examples. In the solution string, every n bits represent one term of the output formula. Let there are n variables in the input examples $\langle x_1, x_2, ..., x_n \rangle$, then if x_i is present in a term, then the i^{th} bit in that term will be 1, otherwise 0.

First we refine this relation R_{k-mDNF} to $\tilde{R}_{k-mDNF}$ as follows: if the solution contains a term which does not has x_1, then that term is outputed as the first term in the solution string.

From the reduction of k-NM colorability to learning problem of monotone k-term-DNF formulae ([PV88]), it is easy to see that there is a direct solution preserving reduction from modified $\tilde{R}_{kNMC}$ to $\tilde{R}_{k-mDNF}$, where projection sequence α is $1, n+1, 2n+1, ..., (k-1)n+1, 2, n+2, ..., (k-1)n+2, ..., n, 2n, ..., kn$ and $|\alpha| = nk$. Therefore, $\tilde{R}_{k-mDNF}$ is universal and R_{k-mDNF} is near-universal. ∎

Theorem 2.11 *Relation R_{kDNF} is near-universal for $k \geq 3$.*

Proof: The solution string in R_{kDNF} is represented by $3nk$ bits, where n is the number of variables in the input examples. In the solution string, every $3n$ bits represent one term of the output formula. First $2n$ bits in a term, represent corresponding $2n$ literals, let say in the order $\langle x_1, \neg x_1, x_2, \neg x_2, ..., x_n, \neg x_n \rangle$, so if x_i is

present in a term, then $(2*(i-1)+1)^{th}$ bit of that term will be 1, otherwise 0. The last n bits in a term contain information on the previous $2n$ bits of the same term as follows:

for $i = 1$ to n
    if $((x_i \vee \neg x_1 \vee ... \neg x_{i-1} \vee \neg x_{i+1} ... \vee \neg x_n)$ is 0)
    then i^{th} bit of last n bits is 0, otherwise 1.

Next, we refine this relation R_{kDNF} to $\tilde{R}_{kDNF}$ as follows: if the solution contains a term, where $(2n+1)^{th}$ bit is 0 in the solution string of that term, then that term is outputed as the first term in the solution string.

Now, it is easy to see the solution preserving reduction from modified $\tilde{R}_{kNMC}$ to $\tilde{R}_{kDNF}$, where projection bits are last n bits of each term in the following order, i.e. $\alpha = 2n+1, 5n+1, ..., 3nk-n+1, 2n+2, 5n+2, ..., 3nk-n+2, ..., 3n, 6n, ..., 3nk$ and $|\alpha| = nk$. Therefore, $\tilde{R}_{kDNF}$ is universal and R_{k-DNF} is near-universal. ∎

2.5 Generic NP-complete Language

Definition 2.7 Generic Complete Language for $\mathcal{NP}$:

$$L_{comp} = \{\langle M, x, 1^n\rangle | \text{ NDTM } M \text{ accepts } x \text{ in at most } n \text{ steps}\}$$

The relation $R_{L_{comp}}$ is: $\langle M, x, 1^n\rangle R_{L_{comp}} w$ iff w is an accepting configuration of M on input x and w has at most n configurations.

Theorem 2.12 *Relation $R_{L_{comp}}$ is universal.*

Proof: We prove the universality of $R_{L_{comp}}$ by showing that $R_{L_{comp}}$ is joinable, couplable and has a building block. The construction of this proof is similar to the proof of universality for the relation witnessing *k-creative sets* ([JY85]) in [AB92].

Let $y = \langle y_1, y_2, ..., y_m \rangle$, where each y_i is $\langle M_i, x_i, 1^{n_i} \rangle$. Define $join_{R_{Lcomp}}(y) = \langle z, \alpha \rangle$, where $z = \langle M, x_1 \# x_2 \# \cdots \# x_m, 1^{p(\sum_{i=1}^{m} n_i)} \rangle$ and NDTM M on input $x_1 \# x_2 \# \cdots \# x_m$ guesses the string $s = w_1 \# w_2 \# \cdots \# w_m$ and accepts iff for every j less than or equal to m, $\langle M_j, x_j, 1^{n_j} \rangle \; R_{L_{comp}} \; w_j$. By suitably choosing $p(.)$, we can ensure that running time of M is enough to simulate all the $M_i's$ for $1 \leq i \leq m$. We assume that the guess string is written in some fixed bit positions in the accepting computation of M, so we can easily compute α.

Define $cpl_{R_{Lcomp}}(\langle M, x, 1^n \rangle, \langle i_1, ..., i_m \rangle, \langle j_1, ..., j_m \rangle) = \langle z, \beta \rangle$, where z is $\langle M_1, x, 1^{q(n)} \rangle$ and NDTM M_1 on input x guesses the string $s = w$ and accepts if $\langle M, x, 1^n \rangle \; R_{L_{comp}}$ w and for all r less than or equal to m, i_r^{th} and j_r^{th} bit of w is different. By suitably choosing $q(.)$, we can ensure that running time of M_1 is enough to simulate M. Here β is identity.

The instance $block_{R_{Lcomp}} = \langle M, x, 1^n \rangle$, where NDTM M on input x guesses a string of length three and accepts in at most n steps iff guessed string is not '000'. ∎

2.6 Conclusion

In this chapter, we have proved some NP-complete problems to be universal – some through properties of their corresponding relations (i.e. join, couple and block) and some through solution preserving reductions from known universal relations. The major goal of the theory of universality is to obtain the proof of NP-completeness through the properties of the relation instead of searching for some suitable known NP-complete problem for the reduction. We have succeeded in getting some results through these properties in the relations, however for others, the standard relations do not seem to possess all the properties.

We have showed R_{kNMC} and R_{kDNF} to be near-universal for $k \geq 3$. For $k = 2$, we could not get the proof because we were trying to prove the universality of R_{2NMC} through solution preserving reduction from standard witnessing relation for

NAE-3SAT (Not All Equal 3SAT ([GJ79])), but standard relation witnessing NAE-3SAT does not seem to be universal.

The defined relation R_{kNMC} is not a natural one, we have put additional information in its solution string in order to prove it near-universal through solution preserving redcution from R_{kNMC}. The standard relation witnessing R_{kNMC} does not seem to be universal.

For further work in this direction, it would be interesting to get the results of universality for R_{kNMC} and R_{kDNF} through characterization of universal relations.

Chapter 3

1-NL Universality

In this chapter, we extend the notion of universality to the complexity class 1-NL. The basic ideas are similar to NL universality ([CSB04]), however it is interesting to apply those ideas here because of the idiosyncrasy of this class due to its one-way read-only input tape. As we stated in the introduction, this class is important because although 1-NL is computationally very weak compared to NL, yet any complete set for it under 1-L reductions is also complete for NL under L reductions. Therefore, if a 1-NL complete set can be shown in L, then the space bounded deterministic and non-deterministic computations for space $\Theta(\log n)$ collapses (i.e. L=NL).

3.1 1-NL Relations

Basic concepts of relations, their associated languages, and solutions can be read from [CSB04].

Definition 3.1 A relation R is a 1-NL relation if it satifies the following conditions:

1. There is a fixed polynomial $p(.)$ such that whenever x is in L_R, $sol\text{-}len_R(x) = p(|x|)$.

2. There is a deterministic logspace machine M_R with two one-way read-only

input tapes, such that, for all (x, y), with x is on one tape and y is on another tape, M_R accepts iff $(x, y) \in R$.

Proposition 3.1 *A language L is in 1-NL iff there is a 1-NL relation R such that $L = L_R$.*

Proof: ($\Rightarrow$) Let L is accepted by a 1-NL machine M. The working of M can be summarized as follows: it reads some bits of the input x from the one-way read-only input tape, and then guesses some bits, and then does some computation on the work-tape. It repeats this process and finally accepts or rejects. Let $y = y_1 y_2 \cdots y_m$, where $y_1, y_2, ..., y_m$ are the bit strings in the same order as they are guessed by M i.e. y_i is guessed before y_{i+1} for all $1 \leq i \leq m-1$). We assume that length of all y_i's are same and fixed. If M accepts x in the above process, then we call y to be the witness for $x \in L$. Now, we define a relation R which contains all such pairs (x, y), for which y is the witness for $x \in L$. Clearly, the length of y is bounded by $p(|x|)$, where $p(.)$ is a polynomial and given (x, y), with x on one one-way read-only input tape, and y on another one-way read-only input tape, a deterministic logspace machine M_R can simulate M on input x, such that all the guess bits of M are already given as y, and it accepts iff M accepts x with the guess bits y. Therefore, the relation R is a 1-NL relation such that $L = L_R$.

($\Leftarrow$) Let R is a 1-NL relation and M_R is a deterministic logspace machine with two one-way read-only input tapes (one for x and one for y), which accepts $(x, y) \in R$. Now, define a languages L, which contains all the $x's$ such that $\exists (x, y) \in R$. We now show that L can be accepted by a 1-NL machine M. The machine M on input x simulates M_R, and whenever M_R needs to read bits from the input tape containing y, guesses those bits, and accepts iff M_R accepts (x, y), here y are the concatenation of all the guess bits by M. Clearly, M accepts L and $L = L_R$. ∎

3.2 Blocks, block-masks, and projections through masking

These definitions for blocks, block-masks and projections throught masking are taken from [CSB04]. We assume that if R is a 1-NL relation, then every solution of length n is the concatenation of a number of equal length blocks. Length of a block in a solution of instance x is $O(log|x|)$. If $x \in L_R$, then *block-len*(x) specifies the length of each block in solutions of x, and further, *block-len*(.) is 1-L computable. (Clearly, *sol-len*(x) is an integral multiple of *block-len*(x)).

The role of *block-mask* is to filter out certain bits from a block. Let l be the length of each block for some instance x. Then the block-mask is also a string of length l over $\{r, d, m_0, m_1\}$. The i^{th} bit of the block-mask specifies what is to be done with the i^{th} bit of a block: r for retaining that bit, d for dropping that bit, m_0 for matching the bit with 0, and m_1 for matching the bit for 1. When a block-mask is applied on a block, the bits of a block are retained or dropped as specified in the mask provided bits those are to be matched as specified by m_0 and m_1 in the mask are indeed matched the block. If the matching is not successful, then the result of applying the mask on the block is an empty sequence. Matched bits, if any, are not retained on masking.

Definition 3.2 (Projection through masking) Let S be a set of solutions where the block length is l, and each solution be of m blocks. Let α be a block-mask. Then the projection of S through α, denoted by $proj_\alpha(S)$ is defined as the set
$\{\alpha(b_1)\alpha(b_2)\cdots\alpha(b_m)|$ Each b_i is a block, and $b_1b_2\cdots b_m \in S\}$

Definition 3.3 (Admissible block-masks) A block-mask α is said to be admissible for a set S of solutions if

1. The length of α is same as that of a block in the solutions.
2. α has at least one r symbol, and

3. If match symbols m_0 or m_1 occur in α, then in every solution in S there will exist at least one block where matching will occur on every match symbol in α.

Proposition 3.2 *For any relation R, a string x, and a block-mask α, such that α is admissible for $sol_R(x)$, then $proj_\alpha(sol_R(x))$ is non-empty iff $x \in L_R$.*

3.3 Universality

Definition 3.4 f: $\Sigma^* \rightarrow \Sigma^*$ is said to be a *solution preserving* reduction from a 1-NL relation Q to a 1-NL relation R if

1. f is computable in 1-L, and
2. For all $x \in \Sigma^*$, there exist some string z and a block-mask α admissible for $sol_R(z)$ such that $f(x) = \langle z, \alpha \rangle$ satisfying $proj_\alpha(sol_R(z)) = sol_Q(x)$.

We assume that the paring function used in above definition is such that its inverse is computable in 1-L.

Proposition 3.3 *If f is a solution preserving reduction from a 1-NL relation Q to a 1-NL relation R, then $L_Q \leq_m^{1\text{-}L} L_R$ via an f', which is an inverse pairing function of f.*

Definition 3.5 A 1-NL relation R is 1-NL universal if for every 1-NL relation Q, there is a solution preserving reduction of Q to R.

Proposition 3.4 *If R is 1-NL universal, then L_R is 1-NL-complete.*

Definition 3.6 A directed acyclic graph G is said to be listed in topologically sorted order if for any pair of edges (a, b) and (b, c) in G, edge (a, b) is listed before (b, c) ([HIM78]).

TAGAP $= \{\langle G, s, t\rangle \mid G$ is a directed acyclic graph given in topologically sorted order, s and t are two distinguished vertices, and there is a path from s to t in $G\}$.

Proposition 3.5 TAGAP *is 1-NL complete under one-way log-tape reductions as well as NL-complete under two-way log-tape reductions ([HM81]).*

We assume that vertices in the graph are labeled as numbers from 1 to n, when there are n vertices in the graph. A directed acyclic graph G can be represented by labeling the vertices in such a way so that all the edges are from lower to higher labeled vertices. We say G is in lexicographically sorted order, if consecutive edges in G satisfy following:

if (a, b) and (c, d) are consecutive edges in G, then either $c > a$ or ($c = a$ and $d > b$).

We say that the representation of a directed acyclic graph G is in restricted topologically sorted order if all the edges in G are from lower to higher labeled vertices and they are listed in lexicographically sorted order.

Definition 3.7 RTAGAP = $\{ \langle G, s, t\rangle \mid G$ is a directed acyclic graph given in restricted topologically sorted order, s and t are two distinguished vertices, and there is a path from s to t in G$\}$.

Proposition 3.6 *By a logspace transducer, we can obtain an instance y of* RTAGAP *from an instance x of* TAGAP *such that $y \in$* RTAGAP *iff $x \in$* TAGAP.

Proof: Let x has n vertices. From x, we obtain a graph $G = \langle V, E\rangle$ such that $V = \langle 1, 2, ..., n\rangle$, and the edges in G are obtained from edges of x as follows: Each edge has two vertices, starting vertex and ending vertex. We go through the edges of x from left to right and relabel the starting vertex of each edge as follows: If this vertex has already been relabeled by some number before, then relabel it with the same number, otherwise relabel it with the least number starting with 1, that has not been assigned to any vertex till now. Next we repeat the same process for the ending vertices. Since x is in topologically sorted order, so G will also satisfy this property after relabeling and note that, now all the edges in G are from lower labeled vertex to higher labeled vertex. Thus only sorting of these edges is remaining to get

the desired instance y from G.

Note that all these operations can be carried out by an L transducer and the graph y is same as x, only we have relabeled the vertices, therefore $x \in$ TAGAP iff $y \in$ RTAGAP. ∎

Without loss of generality, we assume that s and t are 1 and n respectively. (If they are not, then using a 1-L transducer, we can get another restricted topologically sorted graph G' from G such that G has a path from s to t iff G' has a path from 1 to n.)

Let R_{RTAGAP} is the 1-NL relation for RTAGAP, and it is defined as: $(\langle G, 1, n\rangle, y) \in R_{RTAGAP}$ iff $y = v_1 v_2 \cdots v_n$ where G has n vertices, v_j's are vertex labels ({1,2,...,n}) of G, v_1 is 1 and there is some k such that for all j, $1 \leq j < k$, (v_j, v_{j+1}) is an edge of G and $v_{j+1} > v_j$ (Note that all vertices are labeled with numbers), and $v_k = v_{k+1} = \ldots = v_n = n$. In other words, y gives the labels of some 1–n path in G with the end padded with repetitions of the vertex label n so that y has exactly n vertex labels. We need this padding to make every solution of $\langle G, 1, n\rangle$ to have the same length.

Proposition 3.7 RTAGAP *is 1-NL-complete under one-way log-tape reductions as well as NL-complete under two-way log-tape reductions.*

Proof: From the definition, it is clear that RTAGAP is 1-NL-complete under one-way log-tape reductions (proof is slight modification of theorem 2 of [HM81]). Since TAGAP is NL-complete under two-way log-tape reductions and an instance of TAGAP can be easily transformed into an instance of RTAGAP by a logspace transducer. Therefore, RTAGAP is also NL-complete under two-way log-tape reductions. ∎

Theorem 3.1 *Relation R_{RTAGAP} is 1-NL universal.*

Proof: We prove this by showing solution preserving reduction from every 1-NL relation Q to R_{RTAGAP}.

Let Q be a 1-NL relation and M be a 1-NL machine such that $L(M) = L_Q$. Now first see how M works. On input x, M guesses bit by bit the solution y, such that $(x, y) \in Q$ iff $x \in L_Q$.

Let whenever M guesses a bit, it appears on the configuration in the end. Therefore, from each configuration, there are two moves representing guessed bit 0 and 1. Let l_x be the length of configuration of M on input x. (Note that given x, l_x is 1-L computable.) Let START_x is the starting configuration of M on x and ACCEPT is the unique accepting configuration, so if $x \in L$, then M goes to ACCEPT. These different types of configurations can easily be checked by projection through masking (more details are given in [CSB04]).

Consider a 1-L transducer T. It first outputs all the configurations of M on x as the label of vertices of the graph G_x and then it enumerates all the configurations in lexicographically sorted order, and for each such configuration c_1, it again enumerates all the configurations in lexicographically sorted order such that for each such configuration c_2, if M can move from c_1 to c_2 in one step, then it outputs the pair (c_1, c_2) on the output tape representing an edge of G_x. After outputing all such pairs, it outputs START_x and ACCEPT. (Note that this transducer is almost same as the transducer in the proof of theorem 2 of [HM81].)

Next, T outputs block-mask α, of lenght l_x, is such that it retains the last bit of a block (which is in fact the guessed bit used to arrive at the configuration that corresponds to this block from the previous configuration. Note that block length of $\langle G_x, \text{START}_x, \text{ACCEPT}\rangle$ is l_x).

It is clear from the above that M accepts x using y as a solution iff there is a path from START_x to ACCEPT in G_x, and y is the concatenation of the bits chosen by α. Therefore, T is a solution preserving reduction from Q to R_{RTAGAP}. Hence, R_{RTAGAP} is 1-NL universal. ∎

3.4 Characterization of 1-NL Universality

Definition 3.8 (build operation) A 1-NL relation R is said to have the build operation if there is a 1-L computable function $build_R : 1^* \rightarrow \Sigma^*$ such that for all positive integers n, there exist a string x over the alphabet of Σ, and a block mask α, this block-mask is admissible for $sol_R(x)$, such that $block_R(1^n) = \langle x, \alpha \rangle$ satisfying the following:

$$proj_\alpha(sol_R(x)) = \bigcup_{k=1}^{n-2} \bigcup_{P,|P|=k,P\subseteq\{2,\ldots,n-1\}} \bigcup_{\sigma\in S_k} c_1 \cdot c_{\sigma(p_1)} \cdot \ldots \cdot c_{\sigma(p_k)} \cdot c_n \ldots \cdot c_n$$

In the above, S_k denotes all the permutations in increasing order of k elements (i.e. for all i, $S_k[i] < S_k[i+1]$), p_i denotes the i^{th} element of P (P being a set of k integers), c_i denotes the binary encoding of positive integer i, and finally, in the argument string, at the end there will be exactly as many c_n's so that the total number of c_i's in the argument becomes exactly n. (Here $\cdot$ denotes the string concatenation operation.)

Definition 3.9 (prune operation) A 1-NL relation R is said to have the prune operation if there is a 1-L computable function $prune_R : \Sigma^* \rightarrow \Sigma^*$ such that for all n, and for every lexicographically sorted set of pairs $\{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$, all u_i,v_i's being equal length strings, there exist z and an admissible block-mask β such that

$$prune_R(\langle build_R(1^n), \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}\rangle) = \langle z, \beta \rangle$$

with z and β satisfying

$proj_\beta(sol_R(z)) = \{w \mid w \in proj_\alpha(sol_R(x))$, where $x = \pi_1 \circ build_R(1^n)$ and $\alpha = \pi_2 \circ build_R(1^n)$, π_1 and π_2 are inverse pairing functions, with the property that if w is the blocked projection (with α) of the solution y of $\pi_1 \circ build_R(1^n)$, assuming $y = b_1 b_2 \cdots b_r$, each b_i being a block of y, then there exist no j, $1 \leq j < r$, such that for some i, $1 \leq i \leq k$, $u_i v_i$ is being equal to $\alpha(b_j)\alpha(b_{j+1})\}$

Note that we can replace the first parameter $build_R(1^n)$ by 1^n in $prune_R$ function when the definition is clear.

Proposition 3.8 *R_{RTAGAP} has build and prune operations.*

Proof: For all n, $build_{R_{RTAGAP}}(1^n)$ outputs $\langle G, 1, n\rangle$, where G is a directed acyclic graph of n vertices labeled from 1 to n, such that all possible edges from lower to higher labeled vertices are present in G in lexicographically sorted order (i.e. $E(G) = \{(1,2),(1,3),...,(n-1,n)\}$). This function is clearly 1-L computable and the solutions of this instance $\langle G, 1, n\rangle$ satisfy the definition of build operation.

For prune operation, suppose we want to compute $prune_{R_{RTAGAP}}(\langle build_R(1^n), \{(u_1,v_1),(u_2,v_2),...,(u_k,v_k)\}\rangle)$. Note that the set $S = \{(u_1,v_1),(u_2,v_2),...,(u_k,v_k)\}$ is lexicographically sorted. This function will output $\langle G', 1, n\rangle$ such that G' has n vertices and edges in G' are $E(\pi_1 \circ build_R(1^n)) - S$ (here $E(G)$ denotes the edges of graph G). Since $E(\pi_1 \circ build_R(1^n)) - S$ is in lexicographically sorted order, so $prune_{R_{RTAGAP}}$ is computable by a 1-L transducer. ∎

Theorem 3.2 *A 1-NL relation R is universal iff it has build and prune operations.*

Proof: ($\Leftarrow$) : To show this part, we prove that there is a solution preserving reduction from R_{RTAGAP} to R.

Given an instance of R_{RTAGAP}, say $\langle G, 1, n\rangle$, we can construct an instance of R such that the solution space is same for both. The corresponding instance of R is $prune_R(build_R(1^n), S)$, where S is the set $E(\pi_1 \circ build_{R_{RTAGAP}}) - E(G)$. Since G is in restricted topologically sorted order, so we can easily compute S by a 1-L transducer T_1. Therefore, the composition[1] of T_1 followed by the 1-L transducer computing $prune_R$ gives the solution preserving reduction from R_{RTAGAP} to R.

[1] we require 1-L transducers to be closed under composition, [All88] describes two definitions of 1-L transducers where one is closed and another is not.

($\Rightarrow$) : For the other way, since R is universal, so there are solution preserving reductions f from R_{RTAGAP} to R as well as g from R to R_{RTAGAP}.

$build_R$ can simply be defined as:
Let $build_{R_{RTAGAP}}(1^n) = \langle z_1, \alpha \rangle$ and $f(z_1) = \langle z_2, \beta \rangle$, so $build_R(1^n) = \langle z_2, \alpha \cdot \beta \rangle$.

$prune_R(build_R(1^n), S)$ can be obtained using $prune_{R_{RTAGAP}}$ as follows:
Let $prune_{R_{RTAGAP}}(build_{R_{RTAGAP}}(1^n), S)$ is $\langle z_1, \alpha \rangle$. Now let $f(z_1) = \langle z_2, \beta \rangle$. Therefore, $prune_R(build_R(1^n), S)$ is $\langle z_2, \alpha \cdot \beta \rangle$. (Note that we do not need g in proving this direction.) ∎

3.5 Conclusion

In this chapter, we have shown 1-NL universality. The 1-NL set RTAGAP is shown to be 1-NL universal. RTAGAP is a variant of TAGAP (a natural 1-NL-complete problem) and we have shown a reduction from TAGAP to RTAGAP through L transducer. We could not show that TAGAP is 1-NL universal, so it would be interesting to get either the reduction from TAGAP to RTAGAP through 1-L transducer or a different universality notion for 1-NL so that TAGAP can be shown 1-NL universal. We require 1-L transducers to be closed under composition for the characterization of 1-NL universal relations.

Chapter 4

PCP Universality

As we stated in the introduction, PCP gives a different account for the class NP and it defines a very different natural NP-relation. The notion of universality was introduced for the classical NP-relations, so the motivation of this study is to apply the same technique to the new form of NP-relations as per PCP. In this chapter, we go through the sketch of the PCP theorem and thoroughly analyze the proof string, which will later help us in proving the universality of the relation witnessing 3-SAT as per PCP by showing solution preserving reduction from the standard relation witnessing 3-SAT to it.

4.1 PCP Theorem

The basic definitions of the PCP class and other preliminaries can be read from [Aro94, RS95, DK00]. In this section, we sketch of the proof of the PCP theorem with the intention of thoroughly understanding the structure and content of the proof string. The proof of the PCP theorem is divided into four parts and we will go through each part separately.

Theorem 4.1 [PCP Theorem] $\mathcal{NP} = \text{PCP}(\log n, 1)$

Proposition 4.1 $\text{PCP}(\log n, 1) \subseteq \mathcal{NP}$

4.1.1 $\mathcal{NP} \subseteq \text{PCP}(\log n, poly \log n)$

The verifier V of PCP($\log n, poly \log n$) machine has to verify that for the input x, $\exists y$ such that $\varphi(x, y)$ is satisfiable, where $|y| = p(|x|)$ for some polynomial $p(.)$. The prover provides the assignment $\langle x, y \rangle$ in the form of a table $\hat{A}$ of low degree extension of the assignment function A (A: $[h]^m$ (encoding of variables) $\rightarrow$ {0,1}). Now, the verifier's task is to verify that $\hat{A}$ is indeed a low degree extension of some assignment function A which is a satisfying assignment for $\varphi(x, y)$, and finally the assignment for x provided by the prover is consistent with the input x. First the verifier performs low degree test on the table $\hat{A}$. If the test is successful, then it constructs the weighted arithmetization of the 3-SAT formula $\varphi(x, y)$ and uses the following protocol to verify with high probability that the assignment provided by the table $\hat{A}$ is indeed a satisfying assignment to $\varphi(x, y)$. The weighted arithmetization is obtained using random bits (for details, see [AS92, RS95]).

Protocol:

The weighted arithmetization can be finally restated as:

$$\sum_{<w_1, w_2, ..., w_{4m}> \in [h]^{4m}} Q(w_1, w_2, ..., w_{4m}) = 0$$

For $1 \leq i \leq 4m$ and $\alpha_1, \alpha_2, ..., \alpha_{4m} \in \mathcal{F}$ ($\mathcal{F}$ is the large field containing h), let

$$E_i(x) = \sum_{<w_{i+1}, w_{i+2}, ..., w_{4m}> \in [h]^{4m-i}} Q(\alpha_1, \alpha_2, ..., \alpha_{i-1}, x, w_{i+1}, ..., w_{4m})$$

1. Set $\beta = 0$.
2. For $i = 1$ to $4m$
   - Read $\hat{E}_i(x)$ from the proof string.
   - If $\sum_{x \in [h]} \hat{E}_i(x) \neq \beta$, then REJECT.
   - Choose $\alpha_i \in \mathcal{F}$ at random, and set $\beta = \hat{E}_i(\alpha_i)$.
3. Evaluate $\beta' = Q(\alpha_1, \alpha_2, ..., \alpha_{4m})$, and if $\beta' \neq \beta$, then REJECT.
4. Choose $\alpha \in \mathcal{F}^{m-1}$ at random, and if $\hat{A_0^V}(\alpha) \neq \hat{A_0^P}(\alpha)$, then REJECT.

5. ACCEPT.

In the above protocol, $\hat{A_0^V}$ is the low degree extension of the input assignment function, which the verifier can construct itself from the input x, and $\hat{A_0^P}$ is the low degree extension of the input assignment function provided by the prover.

For the entire protocol to work correctly, for every possible random sequence r, which the verifier is using to construct the arithmetization of the 3-SAT formula $\varphi(x, y)$, the prover provides coefficients of low degree extensions $\hat{E}_i(x)$ of all the polynomials $E_i(x)$ which the verifier can possibly asks.

The structure of proof string given by the prover is as follows:

Table for $\hat{A}$ (low degree extension of the assignment function A)

+

For every possible r (random sequence used to construct the weighted arithmetization of $\varphi(x, y)$), coefficients of low degree extensions $\hat{E}_i(x)$ of the polynomials $E_i(x)$ for all i (Since $E_i(x)$ depends on the random numbers picked by the verifier during the above protocol, so for all combinations of $\alpha_1, \alpha_2, ..., \alpha_{i-1} \in \mathcal{F}$, the proof string contains the coefficients of low degree extensions of all such possible $E_i(x)$)

+

Coefficients of the polynomials for the low degree tests

The bits corresponding to the assignment $\langle x, y \rangle$ can be obtained directly from this proof string as follows. The assignment function A assigns value {0,1} to all the variables. Therefore, all the assignment bits can be obtained from the table for A. Instead of table for A, we have been given the table for $\hat{A}$, which is the low degree extension of A. The table for low degree extension $\hat{f}$ of any function f contains all the bits of the table for f. Therefore, all the bits of A are also present in $\hat{A}$. Note that the assignment bits positions are also fixed once the structure of this proof string is fixed.

4.1.2 Input is Encoded

In the protocol above, the verifier reads the input only to verify that the part of assignment provided by the prover is consistent with the input bits. The verifier checks this by first constructing low degree extension of the input, and then verify it with the prover's assignment of the input bits ($\hat{A_0^P}$) at some constant number of random points. In construction of the low degree extension of the input, the verifier reads all the bits of the input. However, if the input is provided as a table of low degree extension of the input bits, then the total number of bits read from this table is bounded by *poly* $\log n$. In this case, the verifier also has to perform low degree test on the table in order to check that it indeed corresponds to some input. In low degree test, the number of bits required from the table are also bounded by *poly* $\log n$, hence the total number of bits read from the table is bounded by *poly* $\log n$ instead of n bits in the original protocol.

4.1.3 Reducing the Number of Probes

In the proof of $\mathcal{NP} \subseteq$ PCP($\log n$, *poly* $\log n$), the verifier is using $O(\log n)$ random bits and reading *poly* $\log n$ bits from the proof string, which are located in different parts of the proof. Next, we reduce the number of probes so that we read these *poly* $\log n$ bits from constant number of locations in the proof string. We can think of the proof string as a function π, which takes address of the location in the above proof string and outputs the bit {0,1} stored at that location (π: I^d (encoding of address) $\rightarrow$ {0,1}). The prover provides the table of the low degree extension $\hat{\pi}$ of π instead of the above proof string. In addition of the table for $\hat{\pi}$, the prover also provides coefficients of the polynomials $F_{a_0,r}$ for every $a_0 \in \mathcal{Q}^d$ ($\mathcal{Q}$ is the large field containing I) and r ($O(\log n)$ random bits used by the verifier). Details of $F_{a_0,r}$ is given in lecture 9 of [RS95]. Note that once the random bit string r is fixed, then all the probing locations in the proof string are also fixed. Now the verifier randomly picks r and a_0 and obtains the coefficients of the polynomial $F_{a_0,r}$ from the proof string and runs the previous protocol such that whenever it needs $\pi(a_i)$, it uses $F_{a_0,r}(i)$ instead.

After reducing the number of probes, the structure of the proof string is as follows:

Table for $\hat{\pi}$

$+$

For all values of r and $a_0 \in \mathcal{Q}^d$, coefficients of the polynomials $F_{a_0,r}$

$+$

Coefficients of the polynomials for the low degree tests

From the above proof string, the assignment $\langle x, y \rangle$ can be easily obtained and their positions can also be fixed. Since there are fixed positions of all the assignment bits $\langle x, y \rangle$ in the earlier proof string and the new proof string contains all the bits of earlier proof string in the table of $\hat{\pi}$, so we can easily map the locations in the earlier proof string to the locations in the new proof string.

4.1.4 $\mathcal{NP} \subseteq PCP(\log n,\ poly \log \log n)$

The verifier of the $PCP(\log n, poly \log n)$ machine works as follows. It gets a random string $\bar{r}$ of $O(\log n)$ and then based on input and $\bar{r}$, it obtains the query locations $\bar{a}$ in the proof string. For the consistency checking of the input bits with the assignment provided by the prover, it obtains a bit string $\bar{b}$, which are the bits in low degree extension of the input at some random points. Finally based on $\bar{r}$ and input, it queries the proof string at some constant number of locations to read constant number of cells. Let these cells are $\bar{c}_1$, $\bar{c}_2$, ..., $\bar{c}_t$. After this, the verifier checks that the string $z = \langle \bar{r}, \bar{a}, \bar{b}, \bar{c}_1, ..., \bar{c}_t \rangle$, of length $poly \log n$, satisfies a polynomial time predicate (or belongs to the language $L \in \mathcal{P}$).

Using Cook-Levin's theorem, we can obtain a 3-SAT formula φ_r which is satisfiable iff the input string z, which depends on the random string $\bar{r}$, belongs to L. For the satisfiability of φ_r, we can again use the same protocol. For this, the prover provides additional proof strings for the satisfiability proof of φ_r for every $\bar{r}$. Note that

in the recursive step, the input z is not completely known to the verifier in advance, and we do not want it to read all the bits from the proof string because its length is $poly \log n$. As we have already seen how we can check membership proof of the input without reading all the bits of the input using table of low degree extension of the input, so the part of the input bits (all c_i's), which are to be read from the proof string, are provided by the prover as low degree extensions, and the verifier constructs itself low degree extensions of the remaining part of the input. Therefore, the bits read from the proof string are only $poly \log \log n$ and the random bits used are still $O(\log n)$.

The structure of the proof string is as follows:

Tables of low degree extensions of every cell (size $\log |\mathcal{Q}|$) of table $\hat{\pi}$

$+$

Tables of low degree extensions of each $F_{a_0,r}$ (size $kd|I| \log |\mathcal{Q}|$)

$+$

For all values of $\bar{r}$, proof string for φ_r

$+$

Coefficients of the polynomials for low degree tests

From the above proof string, the assignment $\langle x, y \rangle$ can be easily obtained and their positions can also be fixed. Since there are fixed positions of all the assignment bits $\langle x, y \rangle$ in the earlier proof string and the new proof string contains all the bits of earlier proof string in the tables of low degree extension of the cells of $\hat{\pi}$ and $F_{a_0,r}$, so we can easily map the locations in the earlier proof string to the locations in the new proof string.

4.1.5 $\mathcal{NP} \subseteq PCP(poly, 1)$

This protocol, which we describe next, uses many random bits ($O(n^3)$), however number of query bits are bounded by some constant ($O(1)$). Here also the prover

tries to convince the verifier that $\varphi(x,y)$ is satisfiable. Let $\bar{a}$ is the assignment encoded in the proof string provided by the prover. The verifier has to check that $\bar{a}$ is the satisfying assignment for $\varphi(x,y)$, which it does as follows: It arithmetizes each clause C_j of $\varphi(x,y)$ and obtains a polynomial $\hat{C}_j(x,y)$. For example, if C_j is $(x_1 \vee \neg x_2 \vee y_3)$, then $\hat{C}_j(x,y) = (1-x_1)x_2(1-y_3)$. Thus $C_j(\bar{a})$ is true iff $\hat{C}_j(\bar{a}) = 0$. Therefore, the verifier has to check that $\forall\ j,\ \hat{C}_j(\bar{a}) = 0$. Note that we are working with field of size 2 in this protocol.

The verifier uses n random bits (let n is the number of clauses in $\varphi(x,y)$) and obtains a weighted sum of the above condition:

$$\sum_j r_j \hat{C}_j(\bar{a}) = 0 .$$

Proposition 4.2 *If* $\exists j\ \hat{C}_j(\bar{a}) \neq 0$, *then* $Pr_r[\sum_j r_j \hat{C}_j(\bar{a}) \neq 0] = 1/2$.

The weighted sum can be expressed as:

$$\sum_j r_j \hat{C}_j(\bar{a}) = \xi + \sum_i \alpha_i a_i + \sum_{<i,j>} \beta_{ij} a_i a_j + \sum_{<i,j,k>} \gamma_{ijk} a_i a_j a_k \qquad (1)$$

where

- ξ is a constant term (either 0 or 1)
- i, j, k are ranging from 1 to n
- α, β, γ are {0,1} vectors of size n, n^2, n^3 respectively.

The prover provides the assignment $\bar{a}$ in terms of tables T_1, T_2, and T_3 defined as:

1. $T_1 : \{0,1\}^n \rightarrow \{0,1\}$ defined by $T_1(\alpha) = \sum_{i=1}^{n} \alpha_i a_i$;
2. $T_2 : \{0,1\}^{n^2} \rightarrow \{0,1\}$ defined by $T_2(\beta) = \sum_{i,j=1}^{n} \beta_{ij} a_i a_j$;
3. $T_3 : \{0,1\}^{n^3} \rightarrow \{0,1\}$ defined by $T_3(\gamma) = \sum_{i,j,k=1}^{n} \gamma_{ijk} a_i a_j a_k$.

Hence checking of weighted sum requires only 3 bits from the proof string. The verifier also has to perform linearity and consistency testing of these tables, which requires $O(n^3)$ random bits and $O(1)$ query bits. Therefore, total number of random bits used are $O(n^3)$ and query bits are $O(1)$ in the above protocol.

The proof string consists of tables T_1, T_2, T_3 of sizes 2^n, 2^{n^2}, 2^{n^3} bits respectively. The assignment $\bar{a}$ can be easily obtained from the table T_1 and its positions are also fixed.

4.1.6 $\mathcal{NP} \subseteq PCP(\log n, 1)$

The verifier V_3 for $PCP(\log n, 1)$ composes two verifiers V_1 in $PCP(\log n, poly \log \log n)$ and V_2 in $PCP(poly, 1)$. It first uses V_1 to get the string $z = \langle r', a', b', c'_1, c'_2, ..., c'_t \rangle$ of length $poly \log \log n$, which now has to be checked for a polynomial time predicate or membership to the language $L \in \mathcal{P}$. Getting this string z requires $O(\log n)$ random bits. The membership proof of z is now checked by V_2, which uses $O(|z|^3)$ random bits and queries constant number of bits from the proof string. Therefore, total number of random bits used by the verifier V_3 is $O(\log n)$ and query bits are $O(1)$ only.

Note that V_3 also has to check the consistency of the input string z with the input assignment provided by the prover. For this, all $c'_i s$ (cells of the proof string of the verifier V_1) are provided by the prover as tables of size $2^{|c_i|}$ bits in the proof string. These tables represent the linear functions defined by the cell entries.

The structure of the proof string is as follows:

Tables T_1, T_2, T_3 for all possible input strings z

$+$

For each cell in the proof string of V_1, table representing the linear function of the cell entries

Since $\bar{a}$ has fixed positions in the proof string of V_1, so it can be easily obtained from the tables representing the linear functions of the cell entries in the proof string of V_1. Therefore, we can easily fixed $\bar{a}$ positions in the proof string of V_3.

Hence $\mathcal{NP} = PCP(\log n, 1)$.

4.2 PCP Universality

We prove that the relation witnessing 3-SAT defined as per PCP is universal by showing solution preserving reduction from the natural witnessing relation of 3-SAT.

Definition 4.1 The relation $R_{3SAT(PCP)}$ is: $\langle\varphi\rangle\ R_{3SAT(PCP)}\ \langle\Pi\rangle$ iff Π is the proof string that causes the verifier V of $PCP(\log n, 1)$ machine M to accept for every random string (i.e. with probability 1).

Proposition 4.3 *$R_{3SAT(PCP)}$ is an NP-relation.*

Proof: In $PCP(\log n, 1)$ machine, the verifier V uses $O(\log n)$ random bits, hence total number of possible random strings are bounded by some polynomial. Therefore, an NP-machine M can simulate all the random bits and verify that V accepts the proof string Π for all the random bits. Every simulation takes only polynomial amount of time and there are polynomial number of iterations, so total time is bounded by some polynomial in the size of the input. ∎

Proposition 4.4 *Relation $R_{3SAT(PCP)}$ is universal.*

Proof: As we have already seen in the proof of the PCP theorem that assignment bits $\bar{a}$ encoded in the proof string Π can be easily fixed, hence there is a solution preserving reduction from R_{3SAT} to $R_{3SAT(PCP)}$. Therefore, relation $R_{3SAT(PCP)}$ is universal. ∎

4.3 Conclusion

We have proved that the relation witnessing 3-SAT as per PCP is universal by showing the solution preserving reduction from the natural relation witnessing 3-SAT. Our goal was to prove it universal by showing it joinable, couplable and having a building block. The instance building block in $R_{3SAT(PCP)}$ is straightforward and restricted couplability can also be shown, but it does not seem to have the join property directly. Therefore, with the current proof string as defined in this chapter, $R_{3SAT(PCP)}$ does not seem to possess all the three properties. However, one can try to modify the structure of the proof string in such a way, so that it can be shown joinable too.

Bibliography

[AB92] Manindra Agrawal and Somenath Biswas. Universal relations. In *Proc. 7th Structure in Complexity Theory Conference*, pages 207–220, 1992.

[All88] Eric W. Allender. Isomorphisms and 1-L reductions. *Journal of Computer and System Sciences*, 36:336–350, 1988.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, pages 13–22, 1992.

[Aro94] Sanjeev Arora. *Probabilistic Checking of Proofs and Hardness of Approximation.* PhD thesis, University of California, Berkeley, 1994.

[AS92] Sanjeev Arora and S. Safra. Probabilistic checking of proofs: A new characterization of np. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[CK03] Gaurav Chakravorty and Rakesh Kumar. #P universality, 2003. BTech Project Report, IIT-Kanpur.

[CLR90] Thomas T. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[CSB04] Vinay Chaudhary, Anand Kumar Sinha, and Somenath Biswas. Universality for nondeterministic logspace. In *Indo-German Workshop on Algorithms.* IISc, Bangalore, 2004.

[DK00] D.-Z. Du and K. Ko. *Theory of Computational Complexity*, chapter 11. Wiley, NY, 2000.

[FM03] H. Fournier and G. Malod. Universal relations and #P-completeness. *Report*, 2003.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Thoery of NP-Completenss.* W. H. Freeman And Company, NY, 1979.

[HIM78] J. Hartmanis, N. Immerman, and S. Mahaney. One-way log-tape reductions. In *19th Annual IEEE Symposium on Foundations of Computer Science*, pages 65–78, 1978.

[HM81] J. Hartmanis and S. Mahaney. Languages simultaneously complete for one-way and two-way log-tape automata. *SIAM Journal of Computing*, 10(2):383–390, 1981.

[HU67] J. E. Hopcroft and J. D. Ullman. Some results on tape-bounded turing machines. *Journal of the ACM*, 16:168–177, 1967.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal of Computing*, pages 935–938, 1988.

[JY85] Deborah Joseph and Paul Young. Some remarks on witness functions for nonpolynomial and noncomplete sets in np. *Theoretical Computer Science*, 39:225–237, 1985.

[KLPV87] Michael Kearns, Ming Li, Leonard Pitt, and Leslie Valiant. On the learnability of boolean formulae. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, pages 285–295, 1987.

[Kum05] Nitesh Kumar. Hardness of approximation problems and proof of PCP theorem. CS397 Report, IIT-Kanpur, 2005.

[KV94] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.

[Pap95] Christos H. Papadimitriou. *Computational Complexity.* Addison–Wesley, 1995.

[PV88] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[RS95] Jaikumar Radhakrishnan and Sanjeev Saluja. Lecture notes on interactive proof systems. Technical report, TIFR, Mumbai, 1995.

[Sud92] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* PhD thesis, University of California, Berkeley, 1992.